Claims

We claim:

1. A method for performing parallel computation of a Discrete Transform of an input signal x, wherein the method operates in a system comprising P interconnected processors and a corresponding P memory mediums, the method comprising:

receiving the input signal x;

the P interconnected processors executing a preprocess in parallel on the signal x to produce a first intermediate vector y;

the P interconnected processors executing a Fourier Transform on said first intermediate vector y to produce a second intermediate vector a; and

the P interconnected processors executing a post-process in parallel on the second intermediate vector a to produce a result vector v, wherein the result vector v comprises the Discrete Transform of the input signal x;

wherein the Discrete Transform of the signal x is useable in analyzing the signal x.

2. The method of claim 1, further comprising:

storing the Discrete Transform of the signal x after said executing the post-process.

3. The method of claim 1, further comprising:

partitioning the signal x into P ordered local vectors; and

distributing one of the P ordered local vectors respectively to each of the P memory mediums before said executing the preprocess.

4. The method of claim 3,

wherein the Discrete Transform is a Discrete Sine Transform; and

wherein said executing a preprocess in parallel on the signal x to produce a first intermediate vector y comprises:

a first processor of said P processors receiving a first sub-vector of the local vector of a first mirror processor of said P processors;

the first processor receiving a first element of the local vector of a second mirror processor of said P processors; and

the first processor computing a first resultant local vector from the local vector of the first processor, said first sub-vector, and said first element.

5. The method of claim 4, wherein said computing the first resultant local vector comprises:

computing a resultant sub-vector from a sub-vector of the local vector of the first processor and the first sub-vector; and

computing a resultant element from a first element of the local vector of the first processor and said first element;

wherein said first resultant local vector comprises said resultant sub-vector and said resultant element.

6. The method of claim 5,

wherein said first resultant local vector comprises an ordered sequence of vector elements comprising an initial element and a sequence of subsequent elements;

wherein said resultant element comprises said initial element of said ordered sequence of vector elements; and

wherein said resultant sub-vector comprises said sequence of subsequent elements.

7. The method of claim 4, wherein the first processor is the i^{th} processor of said P processors, and the first mirror processor is the $(P-1-i)^{th}$ processor of said P processors.

8. The method of claim 4, wherein the second mirror processor is the $(P-i)^{th}$ processor of said P processors.

9. The method of claim 4 further comprising:

said first mirror processor receiving a first sub-vector of the local vector of the first processor;

said first mirror processor receiving a first element of the local vector of a second processor; and

said first mirror processor computing a second resultant local vector from the local vector of the first processor, said first sub-vector and said first element.

10. The method of claim 4, wherein the input signal x comprises N elements, the method further comprising:

each processor of said P processors computing a respective set of coefficients;

wherein said computing a first resultant local vector comprises computing a plurality of resultant values, wherein each of said plurality of resultant values is computed by:

generating a sum and a difference of a respective element of the local vector of the first processor and a complementary element of the first sub-vector;

multiplying the sum by a corresponding one of said coefficients to generate a first product;

multiplying the difference by a constant value to generate a second product; and

adding the first product and second product to generate the resultant value.

11. The method of claim 3,

wherein the Discrete Transform is a Discrete Sine Transform;

wherein said first processor is an initial processor of said P processors;

wherein said executing a preprocess in parallel on the signal x to produce a first intermediate vector y comprises:

a first processor of said P processors receiving a first sub-vector of the local vector of a first mirror processor of said P processors; and

the first processor of said P processors computing a first resultant local vector from a sub-vector of the local vector of the first processor, said first sub-vector and a first element having a value of zero.

12. The method of claim 1,

wherein the Discrete Transform is a Discrete Sine Transform;

wherein said second intermediate vector a comprises P local vectors corresponding respectively to said P processors, wherein each said local vector comprises real values and imaginary values;

wherein said executing a post-process in parallel on the second intermediate vector a to produce the resultant vector v comprises:

each of said P processors computing a respective sequence of partial sums of the real values from the corresponding local vector;

said P processors performing a scan add of the last partial sum of each of said sequences to produce a respective adjustment value for each said sequence;

each of said P processors adding the respective adjustment value to each partial sum of said respective sequence to generate a respective adjusted sequence of partial sums; and

each of said P processors generating a respective local resultant vector from said imaginary values of said corresponding local vector and said respective adjusted sequence of partial sums;

wherein said resultant vector v comprises said local resultant vectors.

13. The method of claim 12, wherein each said local vector resides in a respective local buffer of the corresponding memory medium, wherein said real values

and imaginary values are generated by said FFT so as to occupy alternating positions in said respective local buffer wherein said alternating positions comprise a first set of positions interleaved with a second set of positions;

wherein said generating the respective local resultant vector from said imaginary values of said corresponding local vector and said respective adjusted sequence comprises:

shifting the imaginary values of said respective local buffer so as to move the real values in said respective local buffer from the first set of positions to the second set of positions; and

storing the adjusted sequence of partial sums in said respective local buffer so as to overwrite the first set of positions.

14. The method of claim 3,

wherein said Discrete Transform is a Discrete Cosine Transform;

wherein said executing a preprocess in parallel on the signal x to produce a first intermediate vector y comprises:

each processor i_p of said P processors with index less than P/2 receiving even-indexed elements from the respective local vectors of processor $2i_p$ and processor $2i_p+1$, and storing said even-indexed elements in first and second halves respectively of a second local buffer corresponding to processor i_p; and

each processor i_p of said P processors with index greater than or equal to P/2 receiving odd-indexed elements from processor $2(P-1-i_p)$ and processor $2(P-1-i_p)+1$, and storing said odd-indexed elements in first and second halves respectively of the second local buffer corresponding to processor i_p;

wherein a union of respective contents of said second buffers comprises said intermediate vector y.

15. The method of claim 14 further comprising:

each processor of said P processors overwriting the corresponding local vector with contents of said corresponding second buffer.

16. The method of claim 1,

wherein said Discrete Transform is a Discrete Cosine Transform;

wherein said second intermediate vector a comprises P local vectors corresponding respectively to said P processors, wherein each of said P local vectors represents a set of complex numbers;

wherein said executing a post-process in parallel on the second intermediate vector a to produce the resultant vector v comprises:

each processor i_p performing an element-wise multiplication of the corresponding local vector and a corresponding coefficient vector to generate a corresponding first local resultant vector; and

said P processors performing an even/odd shuffling operation on said corresponding first local resultant vectors to generate corresponding second local resultant vectors.

17. The method of claim 16, wherein said second local resultant vectors corresponding to a subset of said P processors comprise a sequence of elements, wherein said executing a post-process in parallel on the second intermediate vector a to produce the resultant vector v further comprises:

said subset of said P processors performing a forward cyclic shift on said sequence of elements to generate the result vector v.

18. The method of claim 17, wherein said subset of said P processors comprises processors with indices greater than or equal to P/2.

19. The method of claim 16, wherein said performing an even/odd shuffling operation on said corresponding first local resultant vectors to generate corresponding second local resultant vectors comprises:

each processor i_p of said P processors with index less than P/2 receiving even-indexed elements from the respective first local resultant vectors of processor $2i_p$ and processor $2i_p+1$, and storing said even-indexed elements in first and second halves respectively of the second local buffer corresponding to processor i_p; and

each processor i_p of said P processors with index greater than or equal to P/2 receiving odd-indexed elements from the respective first local resultant vectors of processor $2(P-1-i_p)$ and processor $2(P-1-i_p)+1$, and storing said odd-indexed elements in first and second halves respectively of the second local buffer corresponding to processor i_p;

wherein the respective contents of said second buffers comprise said corresponding second local resultant vectors.

20. The method of claim 19, further comprising:

each processor of said P processors overwriting the corresponding local vector with contents of said corresponding second buffer.

21. A system for performing parallel computation of a Discrete Transform of an input signal x, the system comprising:

P interconnected processors;

P memory mediums, wherein each of the memory mediums is coupled to a corresponding one of the P interconnected processors;

an input for receiving the signal x;

wherein the P interconnected processors are operable to:

execute a preprocess in parallel on the signal x to produce a first intermediate vector;

execute a Fourier Transform on said first intermediate vector to produce a second intermediate vector, wherein said second intermediate vector; and

execute a post-process in parallel on the second intermediate vector to produce a result vector, wherein the result vector comprises the Discrete Transform of the input signal x,

wherein the Discrete Transform of the input signal x is useable in analyzing the input signal x.

22. The system of claim 21, wherein said P interconnected processors are further operable to:

store the Discrete Transform of the signal x after said executing the post-process.

23. The system of claim 21, wherein said P interconnected processors are further operable to:

partition the signal x into P ordered local vectors; and

distribute one of the P ordered local vectors respectively to each of the P memory mediums before said executing the preprocess.

24. The system of claim 23,

wherein the Discrete Transform is a Discrete Sine Transform; and

wherein in said executing a preprocess in parallel on the signal x to produce a first intermediate vector y, a first processor of said P processors is operable to:

receive a first sub-vector of the local vector of a first mirror processor of said P processors;

receive a first element of the local vector of a second mirror processor of said P processors; and

compute a first resultant local vector from the local vector of the first processor, said first sub-vector, and said first element.

25. The system of claim 24, wherein in said computing the first resultant local vector, said first processor of said P processors is operable to:

compute a resultant sub-vector from a sub-vector of the local vector of the first processor and the first sub-vector; and

compute a resultant element from a first element of the local vector of the first processor and said first element;

wherein said first resultant local vector comprises said resultant sub-vector and said resultant element.

26. The system of claim 25,

wherein said first resultant local vector comprises an ordered sequence of vector elements comprising an initial element and a sequence of subsequent elements;

wherein said resultant element comprises said initial element of said ordered sequence of vector elements; and

wherein said resultant sub-vector comprises said sequence of subsequent elements.

27. The system of claim 24, wherein the first processor is the i^{th} processor of said P processors, and the first mirror processor is the $(P-1-i)^{th}$ processor of said P processors.

28. The system of claim 24, wherein the second mirror processor is the $(P-i)^{th}$ processor of said P processors.

29. The system of claim 24, wherein said first mirror processor is operable to:

receive a first sub-vector of the local vector of the first processor;

receive a first element of the local vector of a second processor; and

compute a second resultant local vector from the local vector of the first processor, said first sub-vector and said first element.

30. The system of claim 24, wherein the input signal x comprises N elements, wherein each processor of said P processors is operable to compute a respective set of coefficients;

wherein in said computing a first resultant local vector, said first processor is operable to compute a plurality of resultant values, wherein said first processor is operable to compute each said resultant value by:

generating a sum and a difference of a respective element of the local vector of the first processor and a complementary element of the first sub-vector;

multiplying the sum by a corresponding one of said coefficients to generate a first product;

multiplying the difference by a constant value to generate a second product; and

adding the first product and second product to generate the resultant value.

31. The system of claim 23,

wherein the Discrete Transform is a Discrete Sine Transform;

wherein said first processor is an initial processor of said P processors;

wherein in said executing a preprocess in parallel on the signal x to produce a first intermediate vector y, a first processor of said P processors is operable to:

receive a first sub-vector of the local vector of a first mirror processor of said P processors; and

compute a first resultant local vector from a sub-vector of the local vector of the first processor, said first sub-vector and a first element having a value of zero.

32. The system of claim 21,

wherein the Discrete Transform is a Discrete Sine Transform;

wherein said second intermediate vector a comprises P local vectors corresponding respectively to said P processors, wherein each said local vector comprises real values and imaginary values;

wherein in said executing a post-process in parallel on the second intermediate vector a to produce the resultant vector v:

each of said P processors is operable to compute a respective sequence of partial sums of the real values from the corresponding local vector;

said P processors are operable to perform a scan add of the last partial sum of each of said sequences to produce a respective adjustment value for each said sequence;

each of said P processors is operable to add the respective adjustment value to each partial sum of said respective sequence to generate a respective adjusted sequence of partial sums; and

each of said P processors is operable to generate a respective local resultant vector from said imaginary values of said corresponding local vector and said respective adjusted sequence of partial sums;

wherein said resultant vector v comprises said local resultant vectors.

33. The system of claim 32, wherein each said local vector resides in a respective local buffer of the corresponding memory medium, wherein said real values and imaginary values are generated by said FFT so as to occupy alternating positions in said respective local buffer wherein said alternating positions comprise a first set of positions interleaved with a second set of positions;

wherein said generating the respective local resultant vector from said imaginary values of said corresponding local vector and said respective adjusted sequence comprises:

shifting the imaginary values of said respective local buffer so as to move the real values in said respective local buffer from the first set of positions to the second set of positions; and

storing the adjusted sequence of partial sums in said respective local buffer so as to overwrite the first set of positions.

34. The system of claim 31,

wherein said Discrete Transform is a Discrete Cosine Transform;

wherein in said executing a preprocess in parallel on the signal x to produce a first intermediate vector y:

each processor i_p of said P processors with index less than P/2 is operable to receive even-indexed elements from the respective local vectors of processor $2i_p$ and processor $2i_p+1$, and store said even-indexed elements in a second buffer corresponding to processor i_p; and

each processor i_p of said P processors with index greater than or equal to P/2 is operable to receive odd-indexed elements from processor $2(P-1-i_p)$ and processor $2(P-1-i_p)+1$, and store said odd-indexed elements in the second buffer corresponding to processor i_p;

wherein a union of respective contents of said second buffers comprises said intermediate vector y.

35. The system of claim 34 further comprising:

each processor of said P processors overwriting the corresponding local vector with contents of said corresponding second buffer.

36. The system of claim 21,

wherein said Discrete Transform is a Discrete Cosine Transform;

wherein said second intermediate vector a comprises P local vectors corresponding respectively to said P processors, wherein each of said P local vectors represents a set of complex numbers; and

wherein in said executing a post-process in parallel on the second intermediate vector a to produce the resultant vector v:

each processor i_p is operable to perform an element-wise multiplication of the corresponding local vector and a corresponding coefficient vector to generate a corresponding first local resultant vector; and

said P processors are operable to perform an even/odd shuffling operation on said corresponding first local resultant vectors to generate corresponding second local resultant vectors.

37. The system of claim 36, wherein said second local resultant vectors corresponding to a subset of said P processors comprise a sequence of elements;

wherein in said executing a post-process in parallel on the second intermediate vector a to produce the resultant vector v:

said subset of said P processors are operable to perform a forward cyclic shift on said sequence of elements to generate the result vector v.

38. The method of claim 37, wherein said subset of said P processors comprises processors with indices greater than or equal to P/2.

39. The system of claim 36, wherein in said performing an even/odd shuffling operation on said corresponding first local resultant vectors to generate corresponding second local resultant vectors:

each processor i_p of said P processors with index less than P/2 is operable to receive even-indexed elements from the respective first local resultant vectors of processor $2i_p$ and processor $2i_p+1$, and store said even-indexed elements in a second buffer corresponding to processor i_p; and

each processor i_p of said P processors with index greater than or equal to P/2 is operable to receive odd-indexed elements from the respective first local resultant vectors of processor $2(P-1-i_p)$ and processor $2(P-1-i_p)+1$, and store said odd-indexed elements in the second buffer corresponding to processor i_p;

wherein the respective contents of said second buffers comprise said corresponding second local resultant vectors.

40. The system of claim 39, wherein each processor of said P processors is operable to overwrite the corresponding local vector with contents of said corresponding second buffer.

41. A memory medium comprising program instructions for performing parallel computation of a Discrete Transform, wherein the memory medium is comprised in a system comprising a plurality of interconnected processors, wherein said program instructions are executable by the P interconnected processors to perform:

receiving an input signal x;

executing a preprocess in parallel on the signal x to produce a first intermediate vector y;

executing a Fourier Transform on said first intermediate vector y to produce a second intermediate vector, wherein said second intermediate vector a; and

executing a post-process in parallel on the second intermediate vector a to produce a result vector v, wherein the result vector v comprises the Discrete Transform of the input signal x;

wherein the Discrete Transform of the signal x is useable in analyzing the signal x.

42. The memory medium of claim 41, wherein said program instructions are further executable to perform:

storing the Discrete Transform of the signal x after said executing the post-process.

43. The memory medium of claim 41, where said program instructions are further executable to perform:

partitioning the signal x into P ordered local vectors; and

distributing one of the P ordered local vectors respectively to each of the P memory mediums before said executing the preprocess.

44. The memory medium of claim 43,

wherein the Discrete Transform is a Discrete Sine Transform; and

wherein in said executing a preprocess in parallel on the signal x to produce a first intermediate vector y said program instructions are executable by a first processor of said P processors to perform:

receiving a first sub-vector of the local vector of a first mirror processor of said P processors;

receiving a first element of the local vector of a second mirror processor of said P processors; and

computing a first resultant local vector from the local vector of the first processor, said first sub-vector, and said first element.

45. The memory medium of claim 44, wherein said computing the first resultant local vector comprises:

computing a resultant sub-vector from a sub-vector of the local vector of the first processor and the first sub-vector; and

computing a resultant element from a first element of the local vector of the first processor and said first element;

wherein said first resultant local vector comprises said resultant sub-vector and said resultant element.

46. The memory medium of claim 45,

wherein said first resultant local vector comprises an ordered sequence of vector elements comprising an initial element and a sequence of subsequent elements;

wherein said resultant element comprises said initial element of said ordered sequence of vector elements; and

wherein said resultant sub-vector comprises said sequence of subsequent elements.

47. The memory medium of claim 44, wherein the first processor is the j^{th} processor of said P processors, and the first mirror processor is the $(P-1-j)^{th}$ processor of said P processors.

48. The memory medium of claim 44, wherein the second mirror processor is the $(P-j)^{th}$ processor of said P processors.

49. The memory medium of claim 44, wherein said program instructions are executable by said first mirror processor to perform:

receiving a first sub-vector of the local vector of the first processor;

receiving a first element of the local vector of a second processor; and

computing a second resultant local vector from the local vector of the first processor, said first sub-vector and said first element.

50. The memory medium of claim 44, wherein the input signal x comprises N elements, wherein said program instructions are executable by each processor of said P processors to perform:

computing a respective set of coefficients;

wherein said computing a first resultant local vector comprises computing a plurality of resultant values, and wherein each of said plurality of resultant values is computed by:

generating a sum and a difference of a respective element of the local vector of the first processor and a complementary element of the first sub-vector;

multiplying the sum by a corresponding one of said coefficients to generate a first product;

multiplying the difference by a constant value to generate a second product; and

adding the first product and second product to generate the resultant value.

51. The memory medium of claim 41,

wherein the Discrete Transform is a Discrete Sine Transform;

wherein said first processor is an initial processor of said P processors;

wherein in said executing a preprocess in parallel on the signal x to produce a first intermediate vector y, the program instructions are executable by a first processor of said P processors to perform:

receiving a first sub-vector of the local vector of a first mirror processor of said P processors; and

computing a first resultant local vector from a sub-vector of the local vector of the first processor, said first sub-vector and a first element having a value of zero.

52. The memory medium of claim 41,

wherein the Discrete Transform is a Discrete Sine Transform;

wherein said second intermediate vector a comprises P local vectors corresponding respectively to said P processors, wherein each said local vector comprises real values and imaginary values;

wherein in said executing a post-process in parallel on the second intermediate vector a to produce the resultant vector v, the program instructions are executable by each of said P processors to perform:

computing a respective sequence of partial sums of the real values from the corresponding local vector;

wherein in said executing a post-process in parallel on the second intermediate vector a to produce the resultant vector v, the program instructions are executable by said P processors to perform:

performing a scan add of the last partial sum of each of said sequences to produce a respective adjustment value for each said sequence;

wherein in said executing a post-process in parallel on the second intermediate vector a to produce the resultant vector v, the program instructions are further executable by each of said P processors to perform:

adding the respective adjustment value to each partial sum of said respective sequence to generate a respective adjusted sequence of partial sums; and

generating a respective local resultant vector from said imaginary values of said corresponding local vector and said respective adjusted sequence of partial sums;

wherein said resultant vector v comprises said local resultant vectors.

53. The memory medium of claim 52, wherein each said local vector resides in a respective local buffer of the corresponding memory medium, wherein said real values and imaginary values are generated by said FFT so as to occupy alternating positions in said respective local buffer wherein said alternating positions comprise a first set of positions interleaved with a second set of positions;

wherein said generating the respective local resultant vector from said imaginary values of said corresponding local vector and said respective adjusted sequence comprises:

shifting the imaginary values of said respective local buffer so as to move the real values in said respective local buffer from the first set of positions to the second set of positions; and

storing the adjusted sequence of partial sums in said respective local buffer so as to overwrite the first set of positions.

54. The memory medium of claim 41,

wherein said Discrete Transform is a Discrete Cosine Transform;

wherein in said executing a preprocess in parallel on the signal x to produce a first intermediate vector y:

said program instructions are executable by each processor i_p of said P processors with index less than P/2 to perform:

receiving even-indexed elements from the respective local vectors of processor $2i_p$ and processor $2i_p+1$, and storing said even-indexed elements in a second buffer corresponding to processor i_p; and

said program instructions are executable by each processor i_p of said P processors with index greater than or equal to P/2 to perform:

receiving odd-indexed elements from processor $2(P-1-i_p)$ and processor $2(P-1-i_p)+1$, and storing said odd-indexed elements in a second buffer corresponding to processor i_p;

wherein a union of respective contents of said second buffers comprises said intermediate vector y.

55. The memory medium of claim 54 wherein said program instructions are further executable by each processor of said P processors to perform:

overwriting the corresponding local vector with contents of said corresponding second buffer.

56. The memory medium of claim 41,

wherein said Discrete Transform is a Discrete Cosine Transform;

wherein said second intermediate vector a comprises P local vectors corresponding respectively to said P processors, wherein each of said P local vectors represents a set of complex numbers;

wherein in said executing a post-process in parallel on the second intermediate vector a to produce the resultant vector v:

said program instructions are executable by each processor i_p to perform:

performing an element-wise multiplication of the corresponding local vector and a corresponding coefficient vector to generate a corresponding first local resultant vector; and

said program instructions are executable by said P processors to perform:

performing an even/odd shuffling operation on said corresponding first local resultant vectors to generate corresponding second local resultant vectors.

57. The memory medium of claim 56, wherein said second local resultant vectors corresponding to a subset of said P processors comprise a sequence of elements, wherein in said executing a post-process in parallel on the second intermediate vector a to produce the resultant vector v, said program instructions are further executable by said subset of said P processors to perform:

performing a forward cyclic shift on said sequence of elements to generate the result vector v.

58. The memory medium of claim 57, wherein said subset of said P processors comprises processors with indices greater than or equal to P/2.

59. The memory medium of claim 56, wherein in said performing an even/odd shuffling operation on said corresponding first local resultant vectors to generate corresponding second local resultant vectors:

said program instructions are executable by each processor i_p of said P processors with index less than P/2 to perform:

receiving even-indexed elements from the respective first local resultant vectors of processor $2i_p$ and processor $2i_p+1$, and storing said even-indexed elements in a second buffer corresponding to processor i_p; and

said program instructions are executable by each processor i_p of said P processors with index greater than or equal to P/2 to perform:

receiving odd-indexed elements from the respective first local resultant vectors of processor $2(P-1-i_p)$ and processor $2(P-1-i_p)+1$, and storing said odd-indexed elements in a second buffer corresponding to processor i_p;

wherein the respective contents of said second buffers comprise said corresponding second local resultant vectors.

60. The memory medium of claim 59, wherein said program instructions are further executable by each processor of said P processors to perform:

overwriting the corresponding local vector with contents of said corresponding second buffer.

61. A method for performing parallel computation of a Discrete Sine Transform of an input signal x, wherein the method operates in a system comprising P interconnected processors and a corresponding P memory mediums, the method comprising:

receiving an input signal x;

the P interconnected processors executing a preprocess in parallel on the signal x to produce a first intermediate vector y;

the P interconnected processors executing a Fourier Transform on said first intermediate vector y to produce a second intermediate vector a; and

the P interconnected processors executing a post-process in parallel on the second intermediate vector a to produce a result vector v, wherein the result vector v comprises the Discrete Transform of the input signal x;

wherein the Discrete Sine Transform of the signal x is useable in analyzing the signal x.

62. A method for performing parallel computation of a Discrete Cosine Transform of an input signal x, wherein the method operates in a system comprising P interconnected processors and a corresponding P memory mediums, the method comprising:

receiving an input signal x;

the P interconnected processors executing a preprocess in parallel on the signal x to produce a first intermediate vector y;

the P interconnected processors executing a Fourier Transform on said first intermediate vector y to produce a second intermediate vector a; and

the P interconnected processors executing a post-process in parallel on the second intermediate vector a to produce a result vector v, wherein the result vector v comprises the Discrete Transform of the input signal x;

wherein the Discrete Cosine Transform of the signal x is useable in analyzing the signal x.